Take-Home Assignment for Senior Frontend Developer (Vue 3)

Objective:

The objective of this assignment is to evaluate your ability to design and implement a modern web application using Vue 3, demonstrating your skills in component architecture, state management, and user experience. Additionally, you will use Storybook to document and test components in isolation.

Project Description:

You will build a task management application called "TaskMaster" where users can create, edit, and manage their tasks. The application should be visually appealing, responsive, and include the following features:

Assignment Tasks:

1. **Setup and Environment**
    - Set up a new Vue 3 application using Vue CLI or Vite.
    - Use the Composition API for building components.
    - Configure TypeScript to ensure type safety and enhance code quality.
    - Integrate Storybook to document and test UI components.
2. **Core Features**
    - **Task List View:**
        - Display a list of tasks with details such as title, description, due date, and status (e.g., pending, in progress, completed).
        - Implement filtering and sorting options based on status and due date.
    - **Task Management:**
        - Allow users to add new tasks with the necessary details.
        - Enable editing of existing tasks.
        - Implement task deletion functionality.
        - Provide a way to mark tasks as completed or change their status.
3. **User Interface and Experience**
    - Design a clean and intuitive user interface with a focus on user experience.
    - Ensure the application is fully responsive and works seamlessly across different devices and screen sizes.
    - Implement smooth animations or transitions for interactive elements.
4. **State Management**
    - Use Pinia for state management to handle application state efficiently.
    - Ensure the application state is persisted across page reloads (e.g., using local storage).
5. **Component Documentation with Storybook**
    - Set up Storybook for your project and document key components, such as task cards, task lists, and input forms.

    - Create stories for different states and variations of components (e.g., empty task list, completed task).
    - Use Storybook to visually test components in isolation.
6. **Testing**
    - Write unit tests for key components and functionalities using a testing library such as Cypress Component Testing or Storybook Interaction Testing
    - Ensure good test coverage and demonstrate a solid understanding of testing strategies.
7. **Code Quality and Best Practices**
    - Follow best practices for code organization, modularity, and reusability.
    - Use modern JavaScript (ES6+) features and ensure your code is clean, readable, and well-documented.
    - Implement error handling and loading states where appropriate.
8. **Additional Features (Bonus)**
    - Implement a search functionality to find tasks by title or description.
    - Add user authentication using a service like Firebase Auth or Auth0.
    - Integrate with a backend API to persist tasks on a server (you can use a mock API for demonstration).

Submission Guidelines:

- Submit your code as a GitHub repository link
- Commits should be clear and self explanatory
- Include a README file with:
    - Instructions for setting up and running the application.
    - An overview of your design choices and any libraries or tools you used.
    - Any additional features implemented.

Evaluation Criteria:

- Completeness and correctness of the implemented features.
- Code quality, structure, and use of best practices.
- UI/UX design and responsiveness of the application.
- Effective use of TypeScript for type safety.
- Test coverage and quality of test cases.
- Ability to handle complex state management and data persistence.
- Quality of Storybook documentation and component stories.

Bonus Points:

- Implementation of additional features.
- Use of advanced TypeScript features (e.g., generics, utility types).
- Use of Storybook add-ons for accessibility or testing.

Additional Considerations

- **Component Design**: Use Vue 3's Composition API to build reusable and modular components.
- **Accessibility**: Ensure the application is accessible and follows WAI-ARIA guidelines.
- **Performance**: Optimize the application for performance, including lazy loading of components and efficient rendering.